On Fully-Secure Honest Majority MPC without n^2 Round Overhead

Latincrypt'23. Read at ia.cr/2023/1204

Daniel Escudero[1] Serge Fehr[2]

October 5, 2023

[1]J.P. Morgan AI Research & J.P. Morgan AlgoCRYPT CoE

[2]CWI, Amsterdam

Introduction

Secure Multiparty Computation

- **Multiple** parties $P_1, \ldots, P_n$ have **inputs** $x_1, \ldots, x_n$
- They want to compute a function $y = f(x_1, \ldots, x_n)$
- Only leak the **result** y and nothing else about $x_1, \ldots, x_n$
- Security should hold even in an **adversary** controls t out of the n parties

Honest majority

We assume that $t < n/2$:

- The adversary **corrupts** a **minority** of the parties
- The **majority** of parties are **honest**

Why studying this setting?

- We can achieve **information-theoretic** security

$\rightarrow$ More precisely, **statistical** security

- We can achieve **guaranteed output delivery** (G.O.D.)

$\rightarrow$ Meaning the honest parties get correct output regardless of the corrupt parties' behavior

Metrics of interest

We model the function f as an **arithmetic circuit** over a **finite field** $\mathbb{F}$

Communication complexity

Number of field elements communicated in **total**

Number of rounds

Number of **sequential** message exchanges

Known results

[BTH06; BFO12; GSZ20] show that parties can compute an arithmetic circuit C with G.O.D. (aka **full security**) and:

- **Communication complexity** $O(n|C|)$
- **Round complexity** $O(\mathrm{depth}(C) + n^2)$

OUR FOCUS: Improving the round complexity to $O(\mathrm{depth}(C))$.

Motivation

Efficiency:

- For **small** depth(C), the term n^2 adds **many rounds**
- In **distributed** settings, **large** number of rounds **hurts performance**

We can get $O(\text{depth}(C))$ rounds in other related settings!

- $t < n/3$ and **perfect security**
- So why not here too?

The $t < n/3$ setting

For $t < n/3$ it is known that we can get **perfect security**, $O(\mathrm{depth}(C))$ rounds, and **G.O.D.**, with either these properties:

- Increasing to $\Omega(n^2|C|)$ communication.[a]
- *Or*, retaining $O(n|C|)$ communication but assuming **correlated randomness**

> Can we get a similar result for $t < n/2$ and statistical security?

[a] [AAPP23] showed **very recently** that we can actually get $O(n|C|)$.

Our result

We present an MPC for **honest majority** with the following features:

- Statistical security
- Full security (G.O.D.)
- $O(n|C|)$ communication
- $O(\text{depth}(C))$ **rounds**
- Assumes **correlated randomness**

Ongoing work

$O(\text{depth}(C))$ rounds with $O(n^2|C|)$ communication without correlated randomness

Challenges with existing approaches

A successful framework for building protocols with full security is called **dispute control** [BTH06]:

- Place "**checkpoints**" during the protocol execution
- Perform a **fault detection** step at each checkpoint
- If a fault is detected, find a **pair** of parties in **dispute**
- **Re-run** from the previous **checkpoint**, ensuring the same dispute **cannot** occur again

$$\boxed{\begin{array}{c}\#\text{ re-runs} = \#\text{ pairs} \approx n^2 \\ n^2\text{ extra rounds!}\end{array}}$$

We must avoid re-runs!

Secret-sharing-based MPC

Let $\mathbb{F}$ be a **finite field**.

Linear secret-sharing

For $x \in \mathbb{F}$, we denote $[\![x]\!] = (x_1, \dots, x_n)$ a vector of **shares** of x, so that

- Any set of t shares hides the secret x
- Any set of $t + 1$ shares determines the secret x
- The scheme is **linear**: $[\![x]\!] \pm [\![y]\!] = [\![x \pm y]\!]$

FACT

If the parties have certain correlated randomness then MPC reduces to **reconstructing** certain secret-shared values at every multiplication gate

Approach in $t < n/3$ using correlated randomness

FACT

If the parties have certain correlated randomness then MPC reduces to **reconstructing** certain secret-shared values at every multiplication gate

This is exploited to get $O(\text{depth}(C))$ rounds for $t < n/3$ by:

- Designing a **robust reconstruction** protocol
- Ensuring it requires $O(n)$ total communication

Robust reconstruction

For $t < n/3$, use **error correction**

- Ensures incorrect shares can be filtered out and removed

For $t < n/2$, use **robust secret-sharing** [RB89]

- Shares can be endowed with **additional information** that ensures incorrect shares can be filtered out and removed

Reconstruction with $O(n)$ communication

Reconstructing a secret **naively** takes n^2 communication:

- Every party send their share to every other party

Alternatively, send shares to **one party** who reconstructs and sends result back

- $2n = O(n)$ messages
- What if this party decides not to announce anything?
- How to check that the announced reconstruction is correct?

“Multiple kings” idea [DN07]

Assume $t+1$ secrets $[\![s_0]\!], \ldots, [\![s_t]\!]$ will be reconstructed simultaneously.

For $i = 1, \ldots, n$:

- Compute $[\![r_j]\!] = \sum_{\ell=0}^{t} j^\ell \cdot [\![s_\ell]\!]$
- Reconstruct $[\![r_j]\!]$ towards party P_j
- P_j sends r_j to all parties
- Parties recover $(s_0, \ldots, s_t)$ from $(r_1, \ldots, r_n)$.

Communication: $O(\frac{n^2}{t+1}) = O(n)$ per secret

Why does it work?

$(r_1, \ldots, r_n)$ can be seen as a *“shares”* themselves!

What about the $t < n/2$ setting?

Recall: We use **robust secret-sharing**.

- Each share has some **extra information** needed to rule out incorrect shares

Previous approach does **not** work

- Compute $[\![r_j]\!] = \sum_{\ell=0}^{t} j^{\ell} \cdot [\![s_\ell]\!]$
- Reconstruct $[\![r_j]\!]$ towards party P_j
- P_j sends r_j to all parties
- Parties recover $(s_0, \ldots, s_t)$ from $(r_1, \ldots, r_n)$.

Cannot **rule out** incorrect "shares" from $(r_1, \ldots, r_n)$
since they **lack** the "extra information"!

Our solution

We design a **novel** robust secret-sharing scheme that allows P_j to **learn** the *"extra information"* to send alongside r_j, so that the parties can **recover** $(s_0, \ldots, s_t)$ from $(r_1, \ldots, r_n)$

More precisely

Our scheme allows **robustly** reconstructing $(t+1)n$ secrets with $O(n^3)$ communication

- $O(\frac{n^3}{(t+1)n}) = O(\frac{n^3}{n^2}) = O(n)$ **per secret**

Our scheme can be used to obtain MPC with

- **Statistical** and **full** security
- $O(n|C|)$ **communication**
- $O(\text{depthC})$ **rounds**
- Assuming the parties have **correlated randomness**

Technical details

Sit tight (or look at your phone)

We define the sharing $[\![x]\!]$ for a secret $x \in \mathbb{F}$ to consist of

- **Sharing polynomial** $F_0(\mathsf{X}) \in \mathbb{F}_{\leq t}[\mathsf{X}]$ subject to $F_0(0) = x$,
- **Randomizer polynomials** $F_1(\mathsf{X}), \ldots, F_t(\mathsf{X}) \in \mathbb{F}_{\leq t}[\mathsf{X}]$
- **Key polynomials** $A_0(\mathsf{Y}), \ldots, A_t(\mathsf{Y}) \in \mathbb{F}_{\leq t}[\mathsf{Y}]$, and
- **Checking polynomial** $C(\mathsf{X}, \mathsf{Y}) \in \mathbb{F}_{\leq t, \leq t}[\mathsf{X}, \mathsf{Y}]$ given by

$$C(\mathsf{X}, \mathsf{Y}) = F_0(\mathsf{X}) \cdot A_0(\mathsf{Y}) + F_1(\mathsf{X}) \cdot A_1(\mathsf{Y}) + \cdots + F_t(\mathsf{X}) \cdot A_t(\mathsf{Y})\,. \quad (1)$$

$$C(\mathsf{X},\mathsf{Y}) = F_0(\mathsf{X}) \cdot A_0(\mathsf{Y}) + F_1(\mathsf{X}) \cdot A_1(\mathsf{Y}) + \cdots + F_t(\mathsf{X}) \cdot A_t(\mathsf{Y})$$

Every party P_i is given

$$\begin{cases} \boldsymbol{F}(i) := (F_0(i), F_1(i), \ldots, F_t(i)), \\ \boldsymbol{A}(i) := (A_0(i), A_1(i), \ldots, A_t(i)), \\ C(\mathsf{X}, i). \end{cases}$$

Basic **reconstruction**:

- Every P_i sends $(F_0(i), F_1(i), \ldots, F_t(i))$
- Every receiver P_j verifies that

$$C(i,j) = F_0(i) \cdot A_0(j) + F_1(i) \cdot A_1(j) + \cdots + F_t(i) \cdot A_t(j)$$

$O(n)$ reconstruction

Input: $(t+1) \cdot n$ secrets $(\llbracket x^{(m,\ell)} \rrbracket)$, for $\ell \in \{0, \ldots, t\}$ and $m \in \{0, \ldots, n-1\}$, each given by polynomials $(A(\mathsf{Y}), F^{(m,\ell)}(\mathsf{X}), C^{(m,\ell)}(\mathsf{X}, \mathsf{Y}))$.

Output: Each party P_k learns all $(x^{(m,\ell)})_{m,\ell}$.

Assumption: A functionality $\mathcal{F}_{\text{coin}}$ that distributes a random value r to all parties upon request.

Step 1 (intuition)

For each $m \in \{0, \ldots, n-1\}$, reconstruct $\sum_{\ell=0}^{t} j^{\ell} \cdot [\![x^{(m,\ell)}]\!]$ towards P_j

- This is the "multiple king" idea from [DN07]
- It is too expensive if everyone sends **all** of the "extra information" for each $m \in \{0, \ldots, n-1\}$
- Solution: *Compress* this extra information

Step 1 (details I)

Goal: Each P_j learns $\{F_0^{(m)}(0,j) := \sum_{\ell=0}^{t} j^\ell \mathbf{F}^{(m,\ell)}(0)\}_{m=0}^{n-1}$:

- For $m \in \{0,\ldots,n-1\}$, each P_i computes $\mathbf{F}^{(m)}(i,\mathsf{Z}) = \sum_{\ell=0}^{t} \mathsf{Z}^\ell \mathbf{F}^{(m,\ell)}(i)$, and P_i sends $F_0^{(m)}(i,j)$ to each P_j.
- The parties call $\mathcal{F}_{\text{coin}}$ to obtain $\xi \in \mathbb{F}$.
- For $\ell \in \{0,\ldots,t\}$ and $h \in [t]$, each P_i computes $F_h(i,\mathsf{Z}) = \sum_{m=0}^{n-1} \xi^m F_h^{(m)}(i,\mathsf{Z})$, and sends to each P_j the vector $(F_1(i,j),\ldots,F_t(i,j))$.

Step 1 (details II)

- Each P_j computes, for $i \in [n]$, $F_0(i,j) = \sum_{m=0}^{n-1} \xi^m F_0^{(m)}(i,j)$, and upon receiving $(F_1(i,j), \ldots, F_t(i,j))$ from P_i, P_j checks that
$$\boldsymbol{F}(i,j) \cdot \boldsymbol{A}(j) = \sum_{m=0}^{n-1} \sum_{\ell=0}^{t} \xi^m j^\ell \cdot C^{(m,\ell)}(i,j).$$
- Let $\mathcal{I} \subseteq [n]$ be the set of indexes i's for which the check above did not fail. P_j interpolates $F_0^{(m)}(0,j)$ from $\{F_0^{(m)}(i,j)\}_{i \in \mathcal{I}}$

Step 2 (intuition)

For $m \in \{0, \ldots, n-1\}$, each P_j forwards the reconstructed $F_0^{(m)}(0, j) = \sum_{\ell=0}^{t} j^\ell F^{(m,\ell)}(0)$ to all parties

- This is again as in the "multiple king" idea, but how can each receiver P_k verify the correctness of $\{F_0^{(m)}(0, j)\}_{j \in [n]}$?
- Solution: P_j can also interpolate $(F_1(0, j), \ldots, F_t(0, j))$ from $\{\boldsymbol{F}(i, j)\}_{i \in \mathcal{I}}$, so P_j can relay these to the parties
- Problem: "Compressor" ξ is already known before P_j sends $F_0^{(m)}(0, j)$, P_j can cheat.
- Solution++: Sample a new "compressor" and send new compressed extra information

Step 2 (details I)

- For $m \in \{0, \ldots, n-1\}$, each P_j forwards the reconstructed $F_0^{(m)}(0, j) = \sum_{\ell=0}^{t} j^\ell F^{(m,\ell)}(0)$ to all parties
- The parties call $\mathcal{F}_{\text{coin}}$ to obtain $\omega \in \mathbb{F}$.
- Each P_i computes $F'_h(i, \mathsf{Z}) = \sum_{m=0}^{n-1} \omega^m F_h^{(m)}(i, \mathsf{Z})$ for $h \in [t]$. Then P_i sends $(F'_1(i, j), \ldots, F'_t(i, j))$ to each P_j.

Step 2 (details II)

- Each P_j computes, for $i \in [n]$, $F'_0(i,j) = \sum_{m=0}^{n-1} \omega^m F_0^{(m)}(i,j)$, and upon receiving $(F'_1(i,j), \ldots, F'_t(i,j))$ from P_i, P_j checks that
$$\mathbf{F}'(i,j) \cdot \mathbf{A}(j) = \sum_{m=0}^{n-1} \sum_{\ell=0}^{t} \omega^m j^\ell \cdot C^{(m,\ell)}(i,j).$$
- Let $\mathcal{I} \subseteq [n]$ be the set of indexes i's for which the check above did not fail. P_j interpolates $\mathbf{F}'(\mathsf{X}, j)$ from $(\mathbf{F}'(i,j))_{i \in \mathcal{I}}$.

Step 3 (intuition)

- For each P_j, each P_k receives the "extra information" $(F'_1(0,j), \ldots, F'_t(0,j))$
- Each P_k uses this to verify the received values $\{F_0^{(m)}(0,j)\}_{j\in[n]}$, for $m \in \{0, \ldots, n-1\}$
- Each P_k uses the verified "shares" to reconstruct $F_0^{(m)}(0, \mathsf{Z})$, recovering all $x^{(m,\ell)}$'s

Step 3 (details I)

- Each P_j sends $(F'_1(0,j), \ldots, F'_t(0,j))$ to each P_k.
- Upon receiving these values, each P_k computes $F'_0(0,j) = \sum_{m=0}^{n-1} \omega^m \cdot F^{(m)}(0,j)$ and checks that

$$F'(0,j) \cdot A(j) = \sum_{m=0}^{n-1} \sum_{\ell=0}^{t} \omega^m j^\ell \cdot C^{(m,\ell)}(0,j),$$

  for each $j \in [n]$
- Let $\mathcal{J} \subseteq [n]$ be the set of indexes j's for which the check above did not fail. For each $m \in \{0, \ldots, n-1\}$, P_k interpolates $F_0^{(m)}(0, Z) = \sum_{\ell=0}^{t} x^{(m,\ell)} Z^\ell$ from $(F_0^{(m)}(0,j))_{j \in \mathcal{J}}$, and outputs $(x^{(m,\ell)})_{m,\ell}$.

Homework 1

Verify that the total communication is $O(n^3)$

- So communication per secret is $O(\frac{n^3}{(t+1)n}) = O(n)$

Homework 2

Read **Theorem 1** in the paper for security proof

- We get **statistically secure** MPC for **honest majority** with **G.O.D.**, $O(\text{depth}(C))$ rounds and $O(n|C|)$ communication, in the **preprocessing model**
- We do this via a **novel robust secret-sharing** scheme with **efficient** “authentication forwarding”
- See paper for (more) details: `ia.cr/2023/1204`

Thank you!

[AAPP23] I. Abraham et al. “Detect, Pack and Batch: Perfectly-Secure MPC with Linear Communication and Constant Expected Time”. In: 2023.

[BFO12] E. Ben-Sasson, S. Fehr, and R. Ostrovsky. “Near-Linear Unconditionally-Secure Multiparty Computation with a Dishonest Minority”. In: 2012.

[BTH06] Z. Beerliová-Trubíniová and M. Hirt. “Efficient Multi-party Computation with Dispute Control”. In: 2006.

[DN07] I. Damgård and J. B. Nielsen. “Scalable and Unconditionally Secure Multiparty Computation”. In: 2007.

[GSZ20] V. Goyal, Y. Song, and C. Zhu. "Guaranteed Output Delivery Comes Free in Honest Majority MPC". In: 2020.

[RB89] T. Rabin and M. Ben-Or. "Verifiable Secret Sharing and Multiparty Protocols with Honest Majority (Extended Abstract)". In: 1989.